ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ФОРМА НАВЧАННЯ ДЕННА

КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту
Завідувач кафедри _____О.О. Ємець
(підпис)
«_____»_____ 2020 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО БАКАЛАВРСЬКОЇ РОБОТИ
на тему

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТРЕНАЖЕРУ З ТЕМИ «ЗЛИТТЯ ВПОРЯДКОВАНИХ ПОСЛІДОВНОСТЕЙ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ»

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Кулинич Марина Костянтинівна _____ «___»_____2020 р.
(підпис)
Науковий керівник к.ф.-м.н., проф., Ємець Єлизавета Михайлівна
_____ «___»_____ 2020 р.
(підпис)

ПОЛТАВА 2020 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, одиниці, скорочення, терміни	**Пояснення умовних позначень, символів, одиниць, скорочень, термінів**
Merge Sort	Сортування злиттям.
Merge Sort in Place	Сортування злиттям в середині вихідної послідовності.
Merge Sort Out of Place	Сортування злиттям з використанням допоміжних послідовностей послідовності.
Merge of Ordered Sequences	Злиття впорядкованих послідовностей.
a	Перша з двох вихідних послідовностей у методі «злиття впорядкованих послідовностей», в якій елементи є відсортованими.
b	Друга з двох вихідних послідовностей у методі «злиття впорядкованих послідовностей», в якій елементи є відсортованими.
c	Результуюча послідовність у методі «злиття впорядкованих послідовностей», яка утворюється злиттям вихідних відсортованих послідовностей *a* та *b*. Після злиття послідовність *c* повинна бути відсортованою.

ВСТУП

Актуальність теми пов’язана з тим, що при самостійному опануванні теми «Злиття впорядкованих послідовностей» виникає багато труднощів. Програма-тренажер покликана на покрокове розв’язування прикладу разом зі студентом. Таким способом програма виступає в ролі замінника викладача.

Використання тренажерів доцільне при самостійному опрацьовуванні теми працюючими студентами, студентами-заочниками та дистанційниками, студентами, що мають індивідуальний графік відвідування.

Об’єкт розробки – програма-тренажер.

Предмет розробки – тренажер з теми «Злиття впорядкованих послідовностей».

Мета бакалаврської роботи – створити програму, яка реалізує тренажер з теми «Злиття впорядкованих послідовностей».

Задачі бакалаврської роботи: 1) здійснити огляд та аналіз розробок, подібних до тренажеру з сортування чисел; 2) для прикладу з дистанційного курсу «Алгоритми і структури даних» розробити алгоритм, блок-схему алгоритму та програмну реалізацію; 3) другий приклад створити самостійно; для нього створити алгоритм та програмну реалізацію; 4) описати створений тренажер.

Методи розробки – мова програмування С++, середовище програмування Borland Builder.

Нові практичні розробки – створено новий тренажер з теми «Злиття впорядкованих послідовностей».

Ступінь готовності до використання – тренажер повністю готовий до використання та переданий для впровадження у дистанційний курс «Алгоритми і структури даних» Полтавського університету економіки і торгівлі.

Структура роботи: бакалаврська робота складається з чотирьох частин.

Перша частина – постановка задачі – містить технічне завдання до роботи.

Друга частина – інформаційний огляд – містить характеристику та аналіз візуалізацій сортування злиттям, викладених в мережі інтернет.

Третя частина – теоретична – містить алгоритми двох прикладів тренажеру, блок-схему алгоритму першого прикладу.

Четверта частина – практична – містить опис програмної реалізації тренажеру та інструкції по користуванню тренажером.

1. ПОСТАНОВКА ЗАДАЧІ

В бакалаврській роботі необхідно розробити програму типу «тренажер» з теми «Злиття впорядкованих послідовностей».

Тренажер призначений для дистанційного навчального курсу «Алгоритми та структури даних», тому повинен бути побудований на теоретичному матеріалі з цього курсу.

В якості першого прикладу, на базі якого слід розробити тренажер, взяти приклад з дистанційного курсу «Алгоритми та структури даних».

Другий приклад розробити самостійно.

У програмі передбачити виділення кольором (або іншим способом) поточних елементів для послідовностей, що зливаються, та виділенням кольором (або іншим способом) поточного елемента в послідовності, що утворюється.

Тренажер повинен передбачати пояснення помилок на кожному кроці програми, а також підтвердження істинності програми.

Умова прикладу повинна бути видима в будь-який момент виконання програми.

У програмі повинен бути вікно-заголовок з інформацією про студента-розробника, випускаючу кафедру, університет та рік створення програми.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд візуалізацій

У глобальній мережі не вдалось знайти візуалізацій злиття впорядкованих послідовностей. Тому розглянемо візуалізації сортування злиттям [1, 10-12], оскільки злиття впорядкованих послідовностей використовується саме для цього методу.

1) Візуалізація університету Сан-Франциско.

За посиланням http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html (рис. 2.1) представлено візуалізацію сортування злиттям [1, 10].

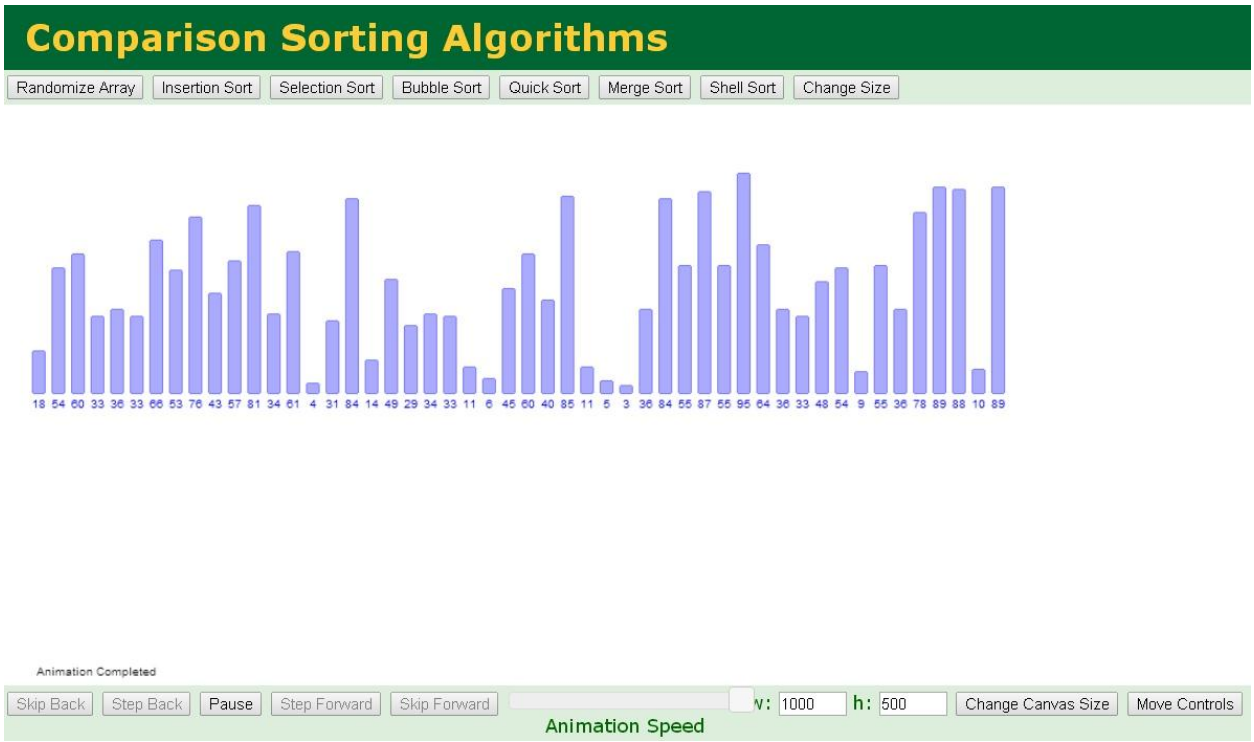

Рисунок 2.1 – Візуалізація університету Сан-Франциско

В анімації числа, що сортуються, зображуються за допомогою прямокутника, висота прямокутника дорівнює числу.

У візуалізації є панелі керування (рис. 2.1).

Перша панель: кнопка *«Randomize array»* генерує числа в послідовності випадковим чином; кнопка *«Merge Sort»* запускає сортування злиттям (сортує числа за зростанням).

Друга панель: повзунок *«Animation Speed»* встановлює швидкість роботи анімації; кнопка *«Skip Back»* повертає до початкової (невідсортованої) послідовності; кнопка *«Skip Forward»* показує відсортовану послідовність.

В цій модифікації методу сортування злиттям візуалізації фази розділення і злиття при сортуванні об’єднані – виконуються одразу в процесі злиття.

2) Візуалізація Трансильванського університету Sapientia.

Ще з однією візуалізацією (хореографічною) можна ознайомитися за посиланням *http://www.youtube.com/watch?v=XaqR3G_NVoo*. Візуалізація створена у вигляді національного німецького танцю [1, 11].

В цій візуалізації фази розділення і злиття при сортуванні також об’єднані – виконуються одразу в процесі злиття.

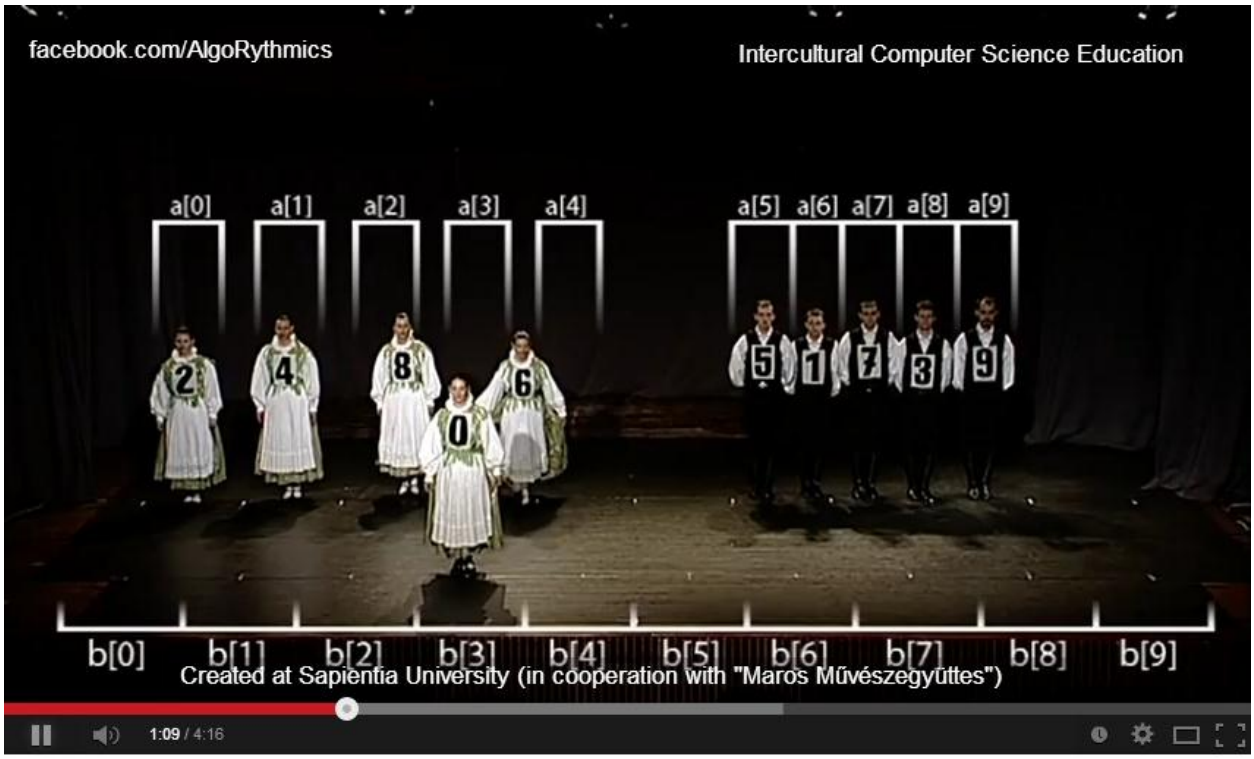

Рисунок 2.2 – Візуалізація Трансильванського університету Sapientia

3) Візуалізація точками.

Ресурс https://youtu.be/zcO8uxg_Spw показує візуалізацію сортувань точками та кольором для 13 алгоритмів, в тому числі, для двох модифікацій сортування злиттям (Merge Sort in Place на 40 секунді, Merge Sort Out of Place) [1, 12].

Рисунок 2.3 – Візуалізація крапками

Пояснимо візуалізацію точками. Кожне число представлено кольоровою точкою.

По горизонтальній осі (вісь x) – вказана позиція елемента в масиві (чим менше індекс, тим ближче до точки відліку – лівого нижнього кута – розташована точка).

Вертикальна вісь (вісь y) відображає значення елемента (чим менше значення, тим ближче до точки відліку розташований елемент).

Відсортований масив буде зображений у вигляді діагональної лінії. Сортування чисел відбувається за зростанням.

Колір в анімації також несе навантаження. Однакові числа мають однаковий колір. Приблизно рівні числа мають схожий відтінок кольору.

2.2. Позитивні аспекти оглянутих візуалізацій

Візуалізація університету Сан-Франциско:

1) є можливість переглядати анімацію покроково, повернутись на крок назад, перейти на крок вперед;

2) є можливість поставити анімацію на паузу;

3) є можливість переглянути роботу анімації різних наборах чисел;

4) є наочність за рахунок вибору представлення чисел у вигляді прямокутників, довжина яких дорівнює числу.

Візуалізація Трансильванського університету Sapientia:

1) творча інтерпретація, що пожвавлює інтерес до вивчаємого матеріалу.

Візуалізація точками:

1) нестандартне представлення чисел дає змогу подивитись на сортування з іншого ракурсу;

2) кожна візуалізація точками виглядає для кожного сортування по-різному (на відміну від візуалізації з використанням прямокутників), що дає можливість краще запам’ятати (за рахунок відмінності) методи сортування.

2.3. Недоліки оглянутих візуалізацій

Візуалізація університету Сан-Франциско:

1) не можна задати свої числа та змінити кількість чисел;

2) щоб розібратись з інтерфейсом анімації слід знати англійську мову;

3) не описана теорія методу сортування.

Візуалізація Трансильванського університету Sapientia:

1) не можна задати свої числа та змінити розмір послідовності;

2) не описана теорія для методу.

Візуалізація точками:

1) візуалізація точками не зразу зрозуміла; в ролику немає пояснень представленню чисел у вигляді точок;

2) не описана теорія для методу.

2.4. Необхідність та актуальність теми

При застосуванні візуалізацій для самостійного навчання виникає декілька проблем:

1) алгоритмів сортування існує багато;

2) для багатьох алгоритмів придумано по декілька модифікацій, тому не завжди візуалізація стовідсотково відповідає тому методу, що вивчається;

3) для одного методу сортування можна налічити багато синонімічних назв;

4) багато візуалізацій містять англомовні назви методів, які ще слід знати;

5) сортувати числа можна за зростанням або за спаданням;

6) деяких допоміжних алгоритмів для сортування чисел не вдається знайти в мережі; зокрема, автору не вдалось знайти візуалізації злиття впорядкованих послідовностей;

7) у візуалізації не використовуються терміни та позначки з теоретичного матеріалу лекції.

Тому переглянута (якщо вона знайдена) візуалізація може не допомогти в освоєнні матеріалу, а навпаки заплутати новачка.

В зв’язку з цим, тренажер, створений саме по матеріалу дистанційного курсу, найкраще допоможе студенту у освоєнні матеріалу.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Алгоритм тренажеру

Побудуємо алгоритм для розробки програмної реалізації тренажеру.

На кожному кроці при правильній відповіді з’являється стверджувальне повідомлення та відбувається перехід на наступний етап.

При хибній відповіді з’являється повідомлення з поясненням помилки. Користувачу знову дається змога надати відповідь. Процес з видачею пояснення помилки продовжується до тих пір, доки надана відповідь не буде вірною.

Приклад 1.

Умова. Є дві послідовності *a={2, 7, 8, 12}* та *b={0, 4, 10}*. Об’єднайте дві послідовності в одну. При цьому утворіть відсортовану за зростанням послідовність, використовуючи метод злиття впорядкованих послідовностей [1, 2].

Крок 1. На екрані: *a={**2**, 7, 8, 12}, b={**0**, 4, 10}, c={ }*. Числа *2* та *0* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={0}*.

При вірній відповіді – «Правильно! Тепер в послідовності *b* поточним стає наступний елемент.».

При помилці – «Помилка! *0 < 2*. Отже, обирається *0*.».

Крок 2. На екрані: *a={**2**, 7, 8, 12}, b={0, **4**, 10}, c={0}*. Числа *2* та *4* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={0, 2}*.

При вірній відповіді – «Правильно! Тепер в послідовності *a* поточним стає наступний елемент.».

При помилці – «Помилка! *2 < 4*. Отже, обирається *2*.».

Крок 3. На екрані: *a={2, **7**, 8, 12}*, *b={0, **4**, 10}*, *c={0, 2}*. Числа *7* та *4* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={0, 2, 4}*.

При вірній відповіді – «Правильно! Тепер в послідовності *b* поточним стає наступний елемент.».

При помилці – «Помилка! *4 < 7*. Отже, обирається *4*.».

Крок 4. На екрані: *a={2, **7**, 8, 12}*, *b={0, 4, **10**}*, *c={0, 2, 4}*. Числа *7* та *10* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={0, 2, 4, 7}*.

При вірній відповіді – «Правильно! Тепер в послідовності *a* поточним стає наступний елемент.».

При помилці – «Помилка! *7 < 10*. Отже, обирається *7*.».

Крок 5. На екрані: *a={2, 7, **8**, 12}*, *b={0, 4, **10**}*, *c={0, 2, 4, 7}*. Числа *8* та *10* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={0, 2, 4, 7, 8}*.

При вірній відповіді – «Правильно! Тепер в послідовності *a* поточним стає наступний елемент.».

При помилці – «Помилка! *8 < 10*. Отже, обирається *8*.».

Крок 6. На екрані: *a={2, 7, 8, **12**}*, *b={0, 4, **10**}*, *c={0, 2, 4, 7, 8}*. Числа *12* та *10* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={0, 2, 4, 7, 8, 10}*.

При вірній відповіді – «Правильно! Послідовність *b* закінчилась. Тому в послідовність *c* внесіть залишок послідовності *a*.».

При помилці – «Помилка! *10* < *12*. Отже, обирається *10*.».

Крок 7. На екрані: *a={2, 7, 8, **12**}*, *b={0, 4, 10}*, *c={0, 2, 4, 7, 8, 10}*. Число *12* виділено. Послідовність *b* закінчилась. Тому в послідовність *c* впишіть (перетягніть) залишок послідовності *a*.».

Правильна відповідь: *c={0, 2, 4, 7, 8, 10, 12}*.

При вірній відповіді – «Правильно! Отже, об’єднали дві послідовності в одну та відсортували! Тренінг завершено!».

При помилці – «Помилка! В послідовності *a* залишився елемент *12*. Отже, обирається *12*.».

Приклад 2.

Умова. Є дві послідовності *a={1, 5, 11}* та *b={-2, 3, 6, 12, 15}*. Об’єднайте дві послідовності в одну. При цьому утворіть відсортовану за зростанням послідовність, використовуючи метод злиття впорядкованих послідовностей.

Крок 1. На екрані: *a={**1**, 5, 11}*, *b={**-2**, 3, 6, 12, 15}*, *c={ }*. Числа *1* та *-2* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={-2}*.

При вірній відповіді – «Правильно! Тепер в послідовності *b* поточним стає наступний елемент.».

При помилці – «Помилка! *-2* < *1*. Отже, обирається *-2*.».

Крок 2. На екрані: *a={**1**, 5, 11}, b={-2, **3**, 6, 12, 15}, c={-2}*. Числа *1* та *3* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={-2, 1}*.

При вірній відповіді – «Правильно! Тепер в послідовності *a* поточним стає наступний елемент.».

При помилці – «Помилка! *1 < 3*. Отже, обирається *1*.».

Крок 3. На екрані: *a={1, **5**, 11}, b={-2, **3**, 6, 12, 15}, c={-2, 1}*. Числа *5* та *3* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={-2, 1, 3}*.

При вірній відповіді – «Правильно! Тепер в послідовності *b* поточним стає наступний елемент.».

При помилці – «Помилка! *3 < 5*. Отже, обирається *3*.».

Крок 4. На екрані: *a={1, **5**, 11}, b={-2, 3, **6**, 12, 15}, c={-2, 1, 3}*. Числа *5* та *6* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={-2, 1, 3, 5}*.

При вірній відповіді – «Правильно! Тепер в послідовності *a* поточним стає наступний елемент.».

При помилці – «Помилка! *5 < 6*. Отже, обирається *5*.».

Крок 5. На екрані: *a={1, 5, **11**}, b={-2, 3, **6**, 12, 15}, c={-2, 1, 3, 5}*. Числа *11* та *6* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={-2, 1, 3, 5, 6}*.

При вірній відповіді – «Правильно! Тепер в послідовності *b* поточним стає наступний елемент.».

При помилці – «Помилка! *6 < 11*. Отже, обирається *6*.».

Крок 6. На екрані: *a={1, 5, **11**}, b={-2, 3, 6, **12**, 15}, c={-2, 1, 3, 5, 6}*. Числа *11* та *12* виділені. Порівняйте виділені числа та менше з них впишіть (перетягніть) в послідовність *c*.

Правильна відповідь: *c={-2, 1, 3, 5, 6, 11}*.

При вірній відповіді – «Правильно! Послідовність *a* закінчилась. Тому в послідовність *c* внесіть залишок послідовності *b*.».

При помилці – «Помилка! *11 < 12*. Отже, обирається *11*.».

Крок 7. На екрані: *a={1, 5, 11}, b={-2, 3, 6, **12**, 15}, c={-2, 1, 3, 5, 6, 11}*. Число *12* виділено. Послідовність *a* закінчилась. Тому в послідовність *c* впишіть (перетягніть) залишок послідовності *b*.».

Правильна відповідь: *c={-2, 1, 3, 5, 6, 11, 12, 15}*.

При вірній відповіді – «Правильно! Отже, об’єднали дві послідовності в одну та відсортували! Тренінг завершено!».

При помилці – «Помилка! В послідовності *b* залишилися елементи *12, 15*. Отже, вносимо в *c 12, 15*.».

3.2. Блок-схема алгоритму

На рис. 3.1-3.7 зображена блок-схема алгоритму для прикладу 1.

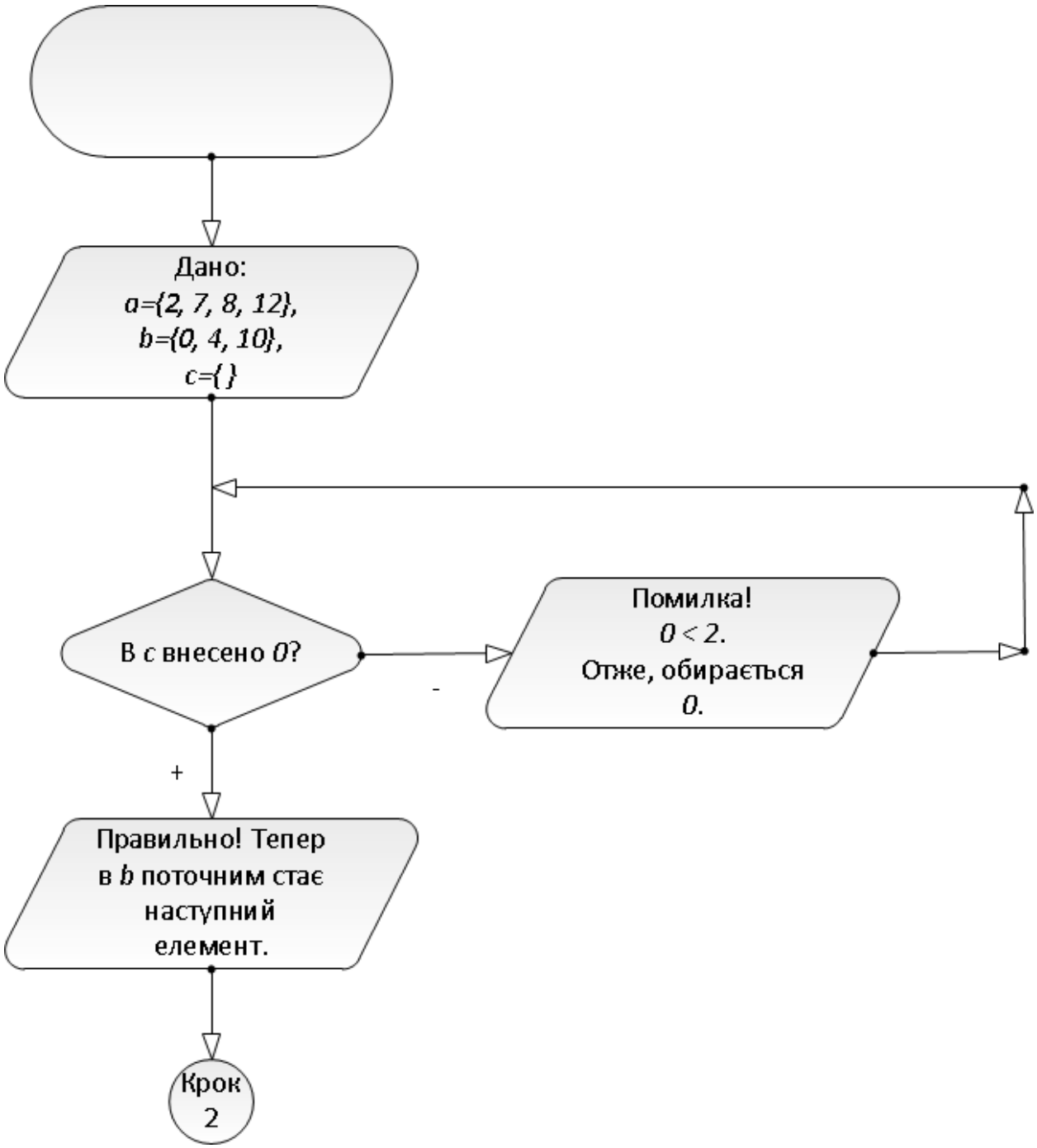

Рисунок 3.1 – Блок-схема алгоритму

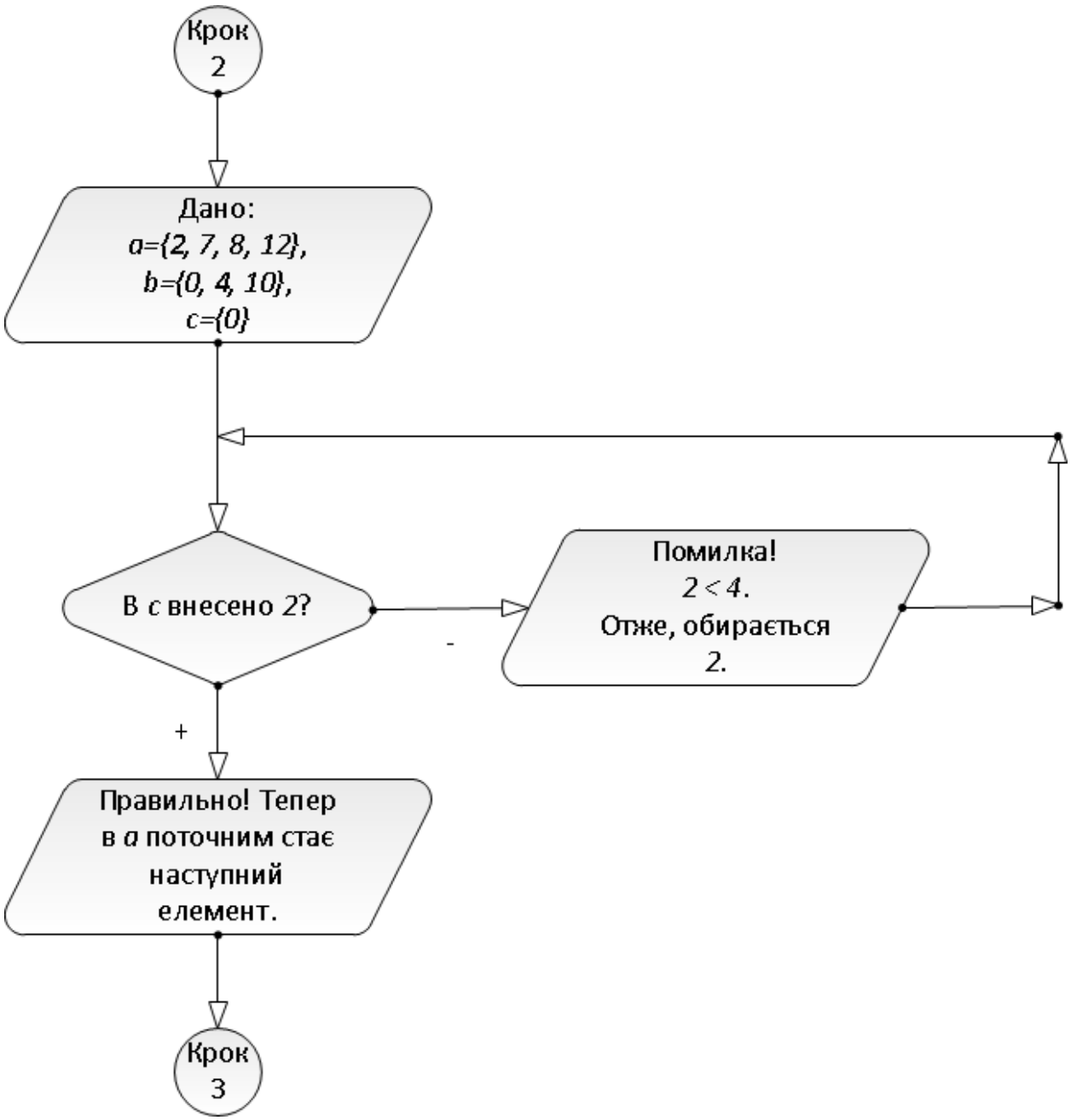

Рисунок 3.2 – Блок-схема алгоритму

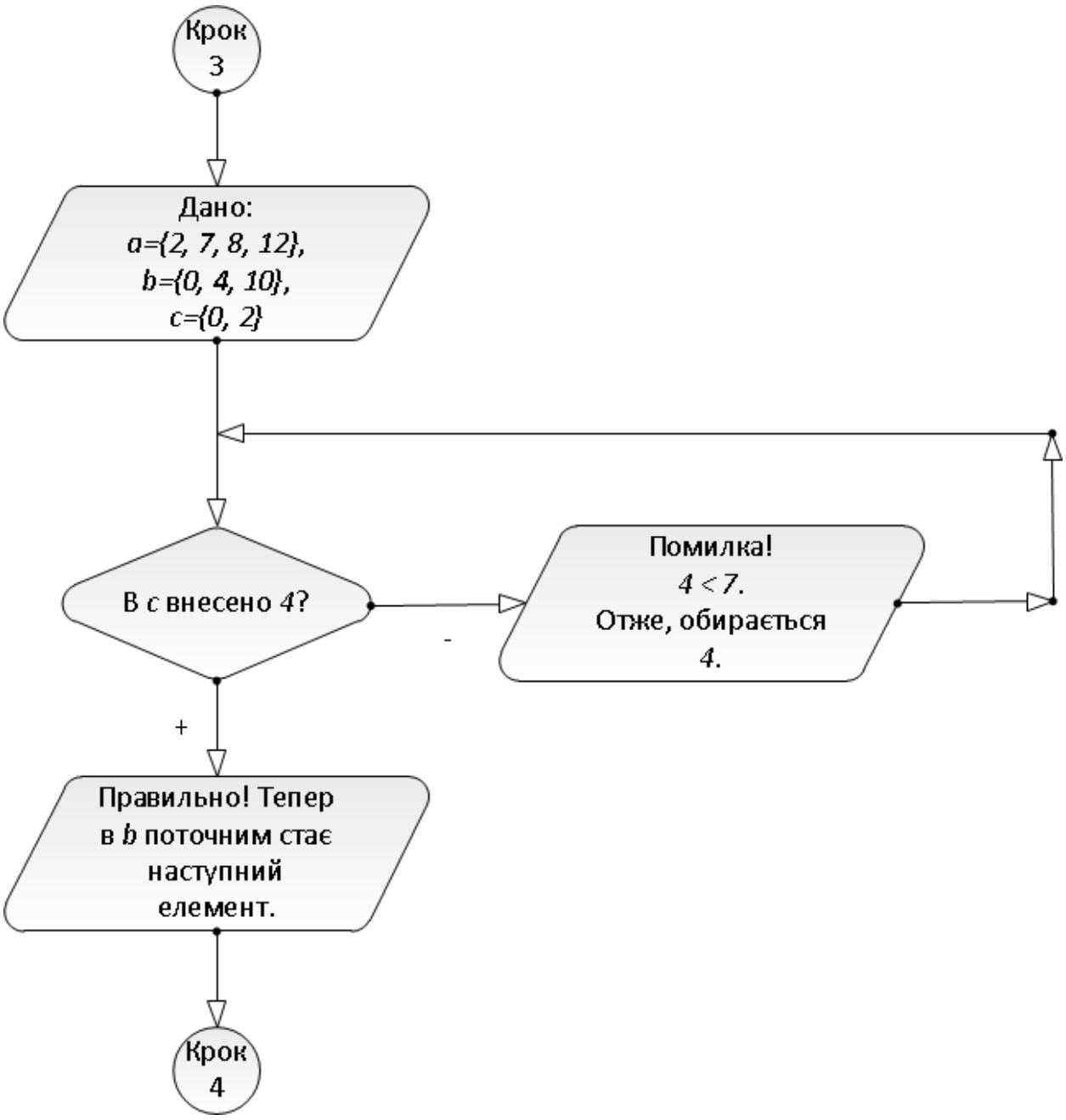

Рисунок 3.3 – Блок-схема алгоритму

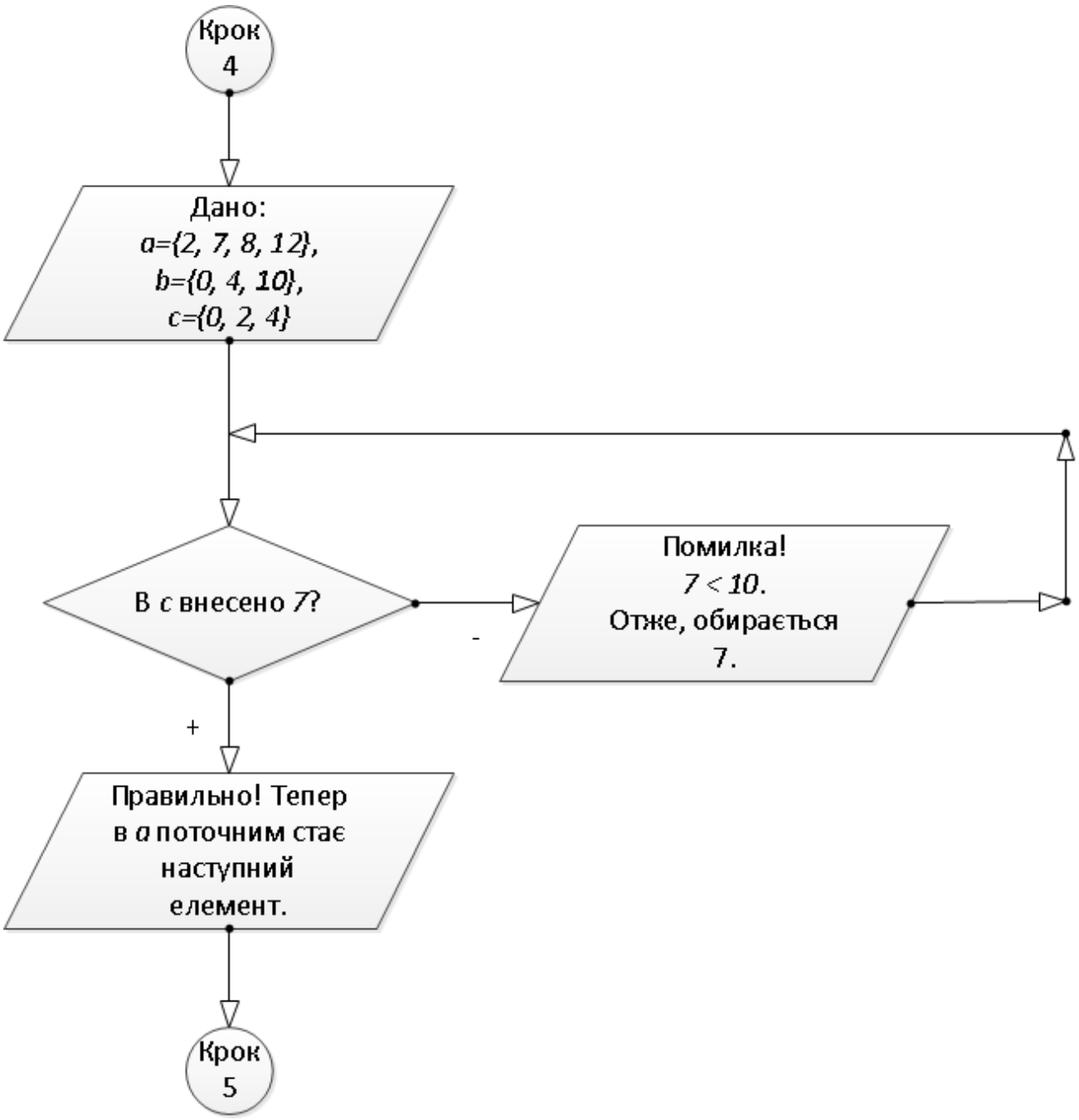

Рисунок 3.4 – Блок-схема алгоритму

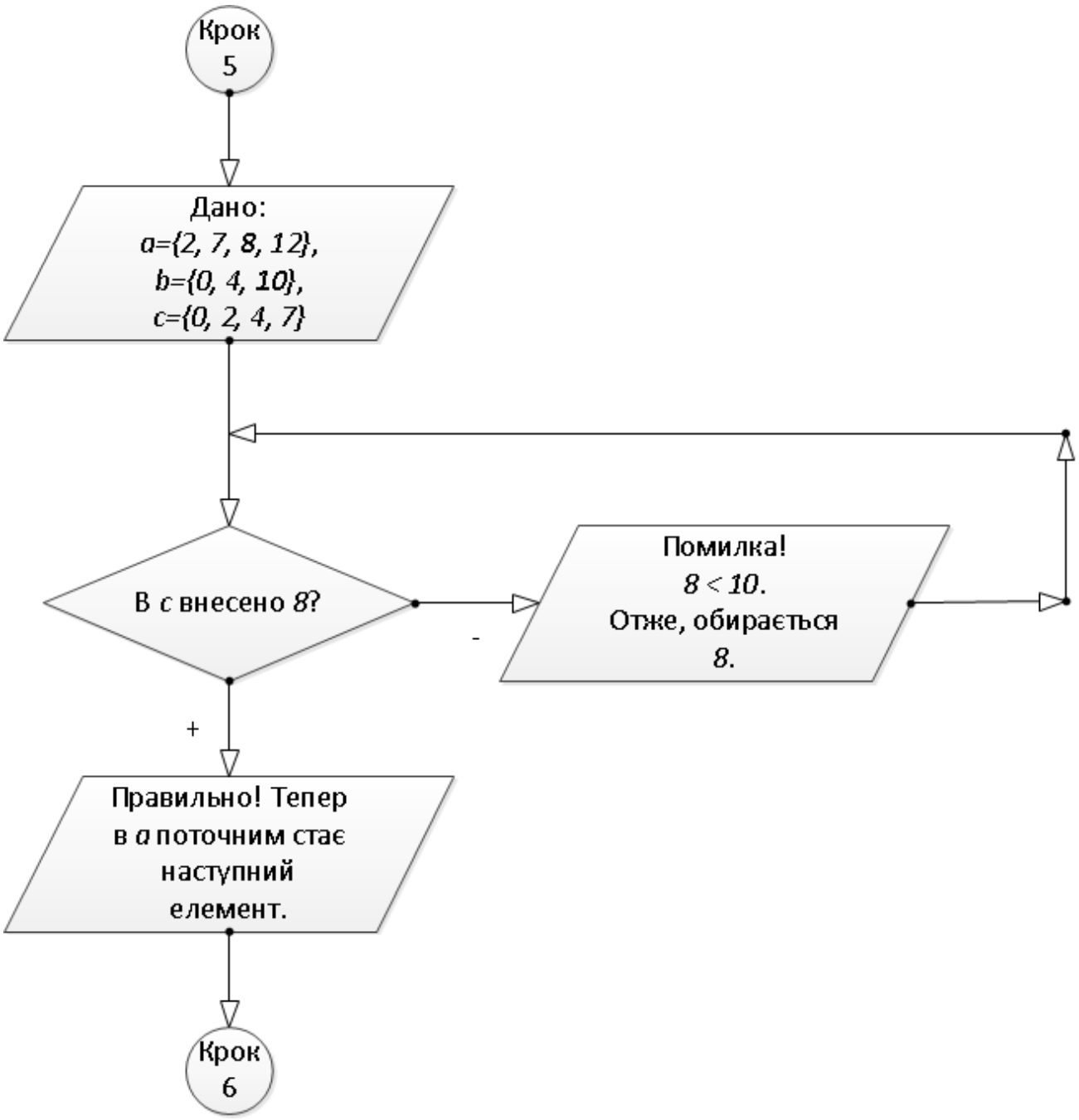

Рисунок 3.5 – Блок-схема алгоритму

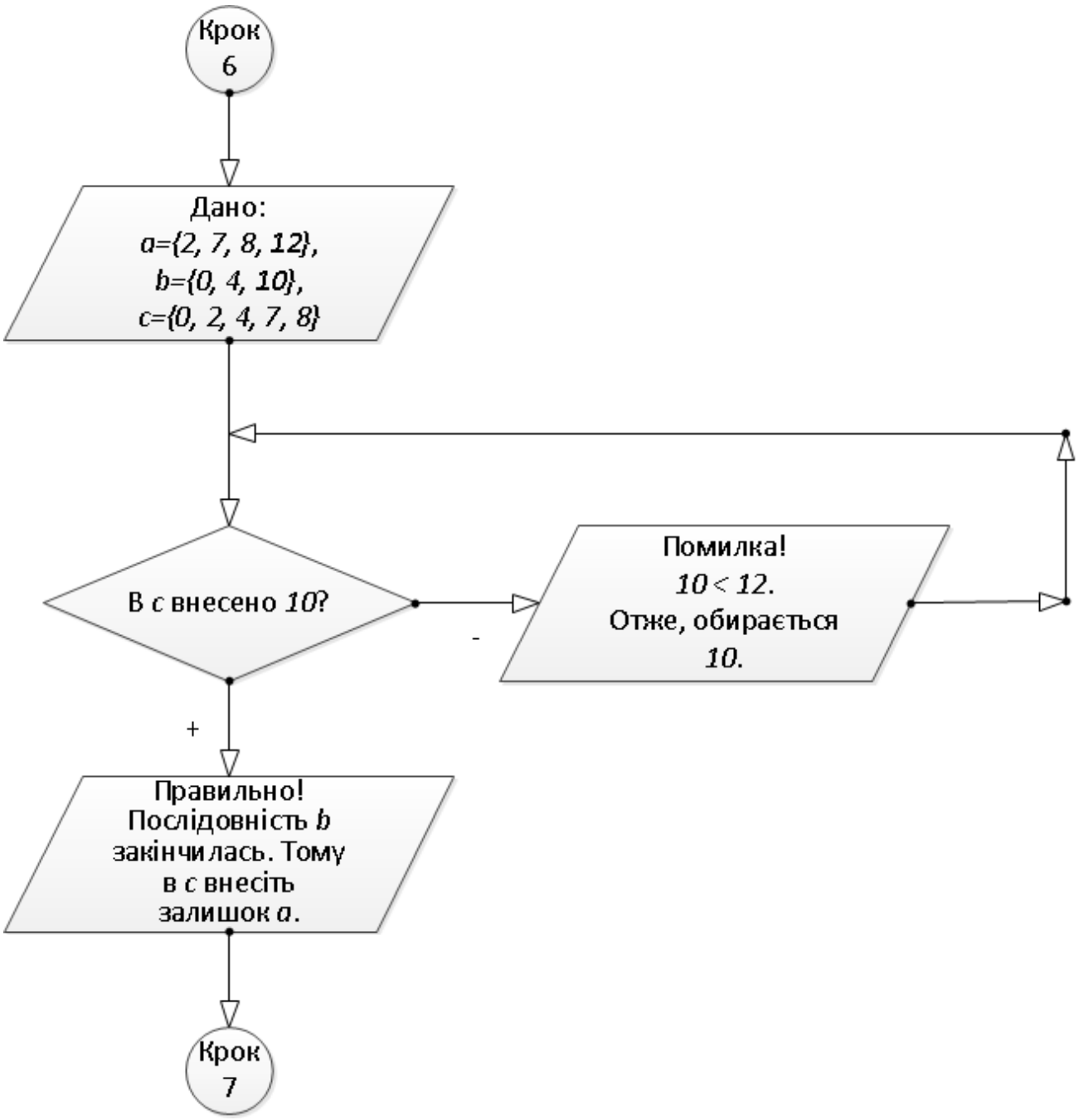

Рисунок 3.6 – Блок-схема алгоритму

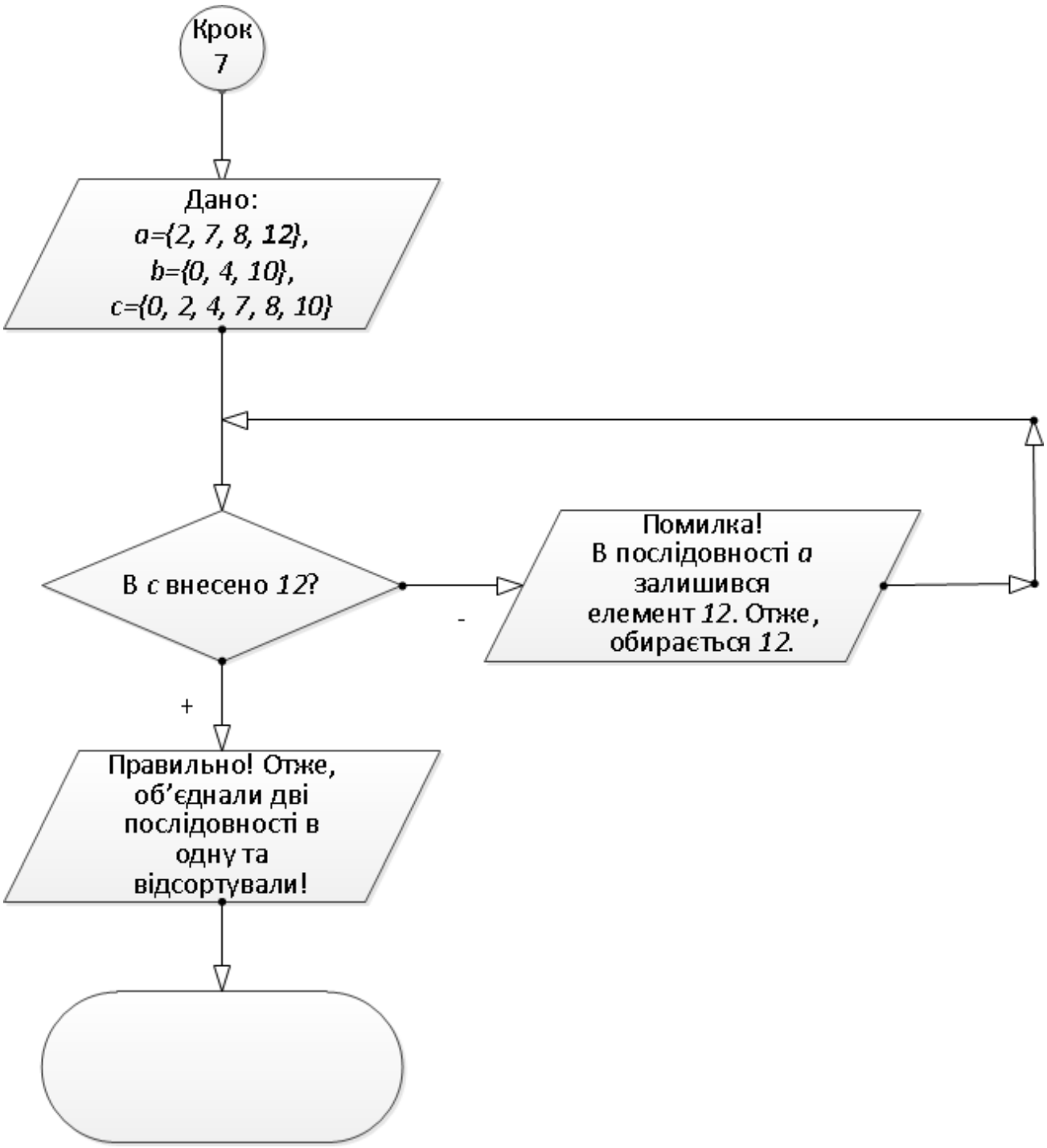

Рисунок 3.7 – Блок-схема алгоритму

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис програмної реалізації

Програма була створена у середовищі Borland Builder з використанням мови програмування C++.

Розглянемо процес створення програми на прикладі 1, етап 1 (рис. 4.1).

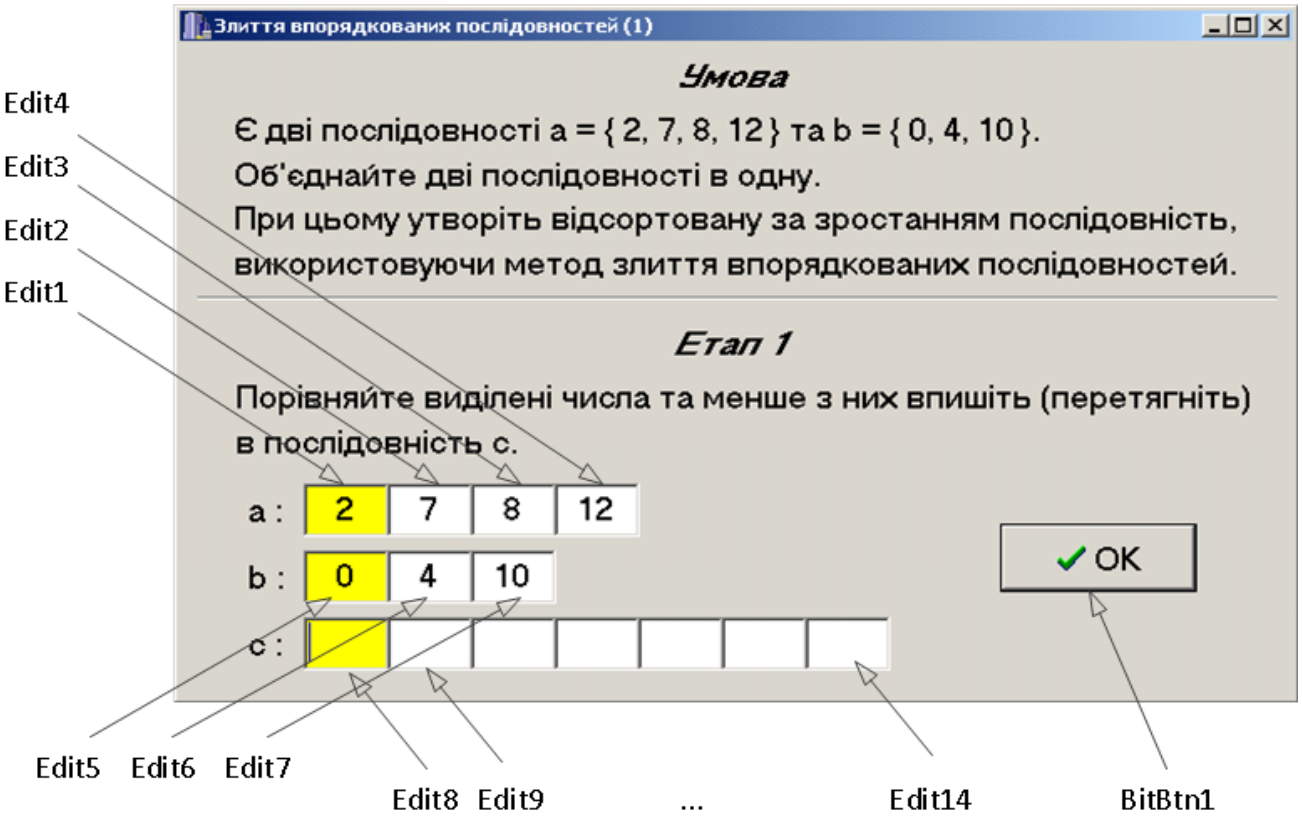

Рисунок 4.1 – Структура форми

Для відображення елементів послідовностей були обрані текстові елементи Edit (див. рис. 4.1).

Для зміни кольору фону в компоненті Edit на жовтий використовувалась властивість Color, якій було задано значення clYellow.

Виходячи зі специфіки прикладу, у програмі доцільно або вводити числа або перетягувати їх з потрібних послідовностей.

Компоненти, з яких перетягуємо інформацію, – це Edit1 або Edit5. Для них властивість DragMode отримала значення dmAutomatic.

Компонент, куди перетягуємо інформацію, – це Edit8. Для нього було зініційовано дві події: OnDragOver, OnDragDrop

Перша подія (OnDragOver) говорить, з яких компонентів можна перетягувати інформацію:

```
void __fastcall TForm2::Edit8DragOver (TObject *Sender,
TObject *Source, int X, int Y, TDragState State, bool
&Accept)
{
  Accept = Source->ClassNameIs("TEdit");
}
```

В даному випадку тільки з компонентів типу Edit.

Друга подія (OnDragDrop) зазначає, що відбувається, коли процес перетягування здійснився:

```
void __fastcall TForm2::Edit8DragDrop (TObject *Sender,
TObject *Source, int X, int Y)
{
  if (Source==Edit1)
    Edit8->Text=Edit1->Text;

  if (Source==Edit5)
    Edit8->Text=Edit5->Text;
}
```

В даному випадку, якщо компонент з якого перетягуємо, це компонент Edit1 або Edit5, то їх вміст копіюємо в компонент Edit8.

Для перевірки вірності відповіді користувач натискає кнопку «OK». Для цього був написаний код:

```
void __fastcall TForm2::BitBtn1Click(TObject *Sender)
{
   if ((Edit8->Text==Edit5->Text)||(Edit8->Text=="0"))
   {
     MessageDlg("Правильно!", mtInformation,
TMsgDlgButtons() << mbOK, 0);
     MessageDlg("Тепер в послідовності b поточним стає
наступний елемент.", mtWarning, TMsgDlgButtons() << mbOK,
0);
     Form3->Show();
     Form3->Edit9->SetFocus();
     Form2->Hide();
   }
   else
   {
     MessageDlg("Помилка!\n0 < 2.\nОтже, обирається 0.",
mtError, TMsgDlgButtons() << mbOK, 0);
     Edit8->SetFocus();
   }
}
```

Для гарного вигляду числа в текстових полях були вирівняні по центру за допомогою пробілів. Тому в програмному коді перевіряється, чи правильна відповідь (число) була введена з клавіатури, чи перетягнута з компонента.

Якщо відповідь правильна, то 1) за допомогою функції MessageDlg виводиться підтверджуюче повідомлення, 2) виводиться повідомлення, який

елемент стає поточним, 3) відкривається наступне вікно, в якому ставлять курсор в потрібний компонент, 4) приховується поточне вікно.

Якщо відповідь хибна, то 1) за допомогою функції MessageDlg виводиться пояснення помилки, 2) курсор ставиться в компонент з помилковою відповіддю.

Повний програмний код для першого прикладу винесено у додаток Б.

4.2. Інструкція по роботі з програмою

На рис. 4.2 показано перше вікно програми. По натисненню кнопки «Тренінг» відбувається перехід до першого кроку тренажера (рис. 4.3).

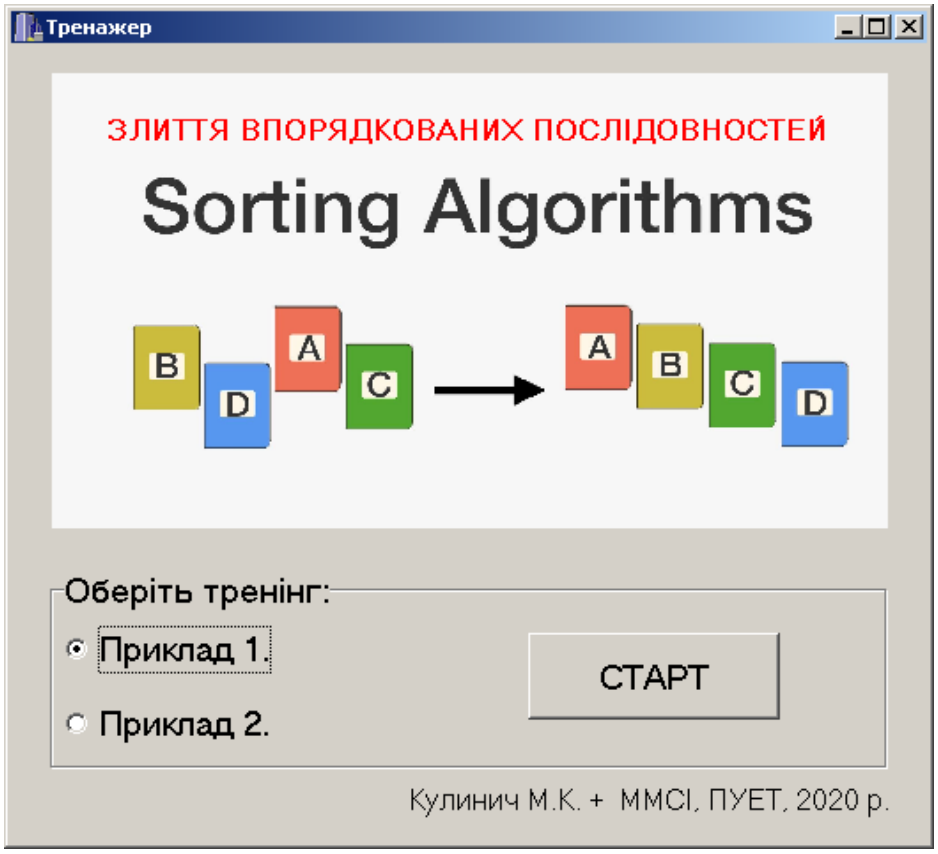

Рисунок 4.2 – Головне вікно програми

У вікні жовтим кольором виділено поточні елементи послідовностей, що зливаються. Їх слід порівняти та менше з них записати в поточну клітинку третьої послідовності. Ця клітинка також виділені жовтим кольором.

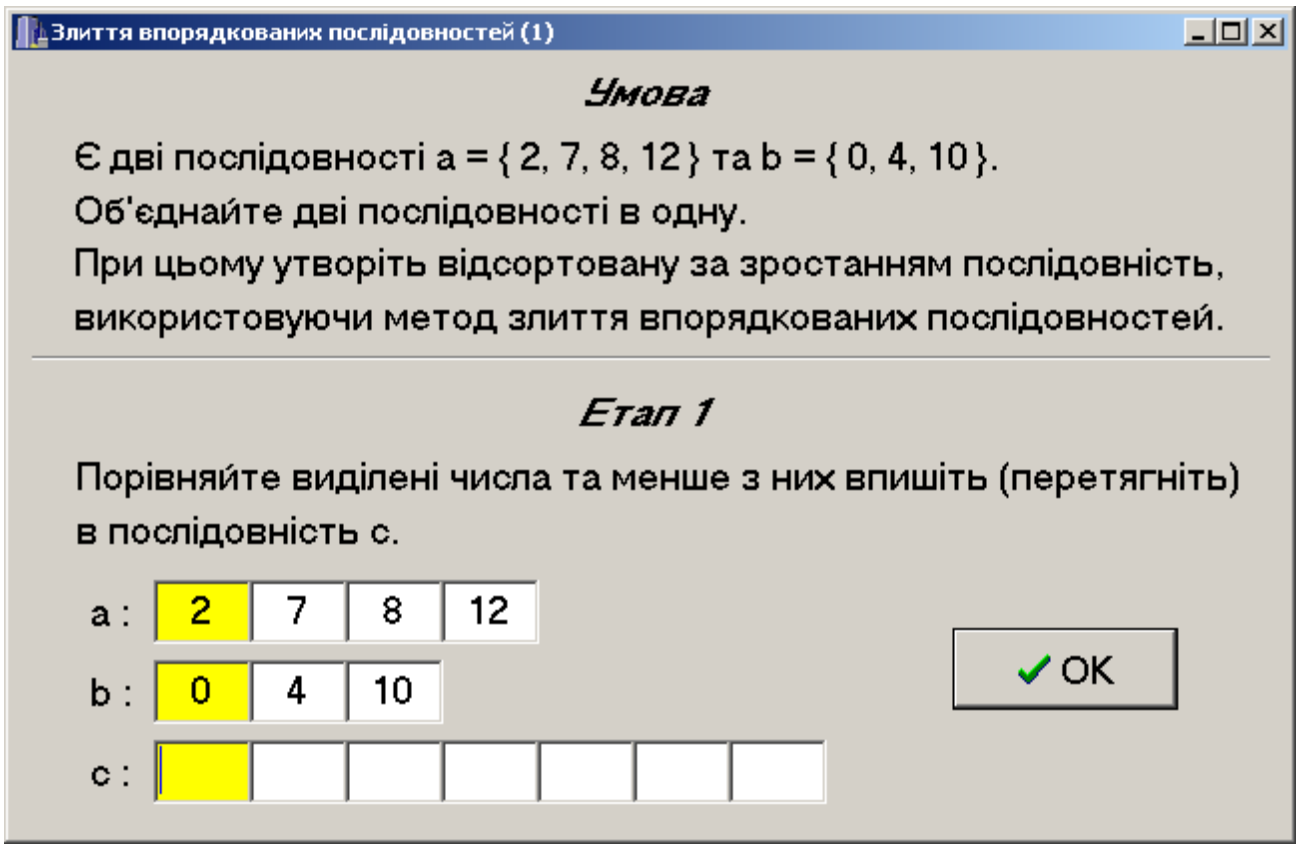

Рисунок 4.3 – Етап 1

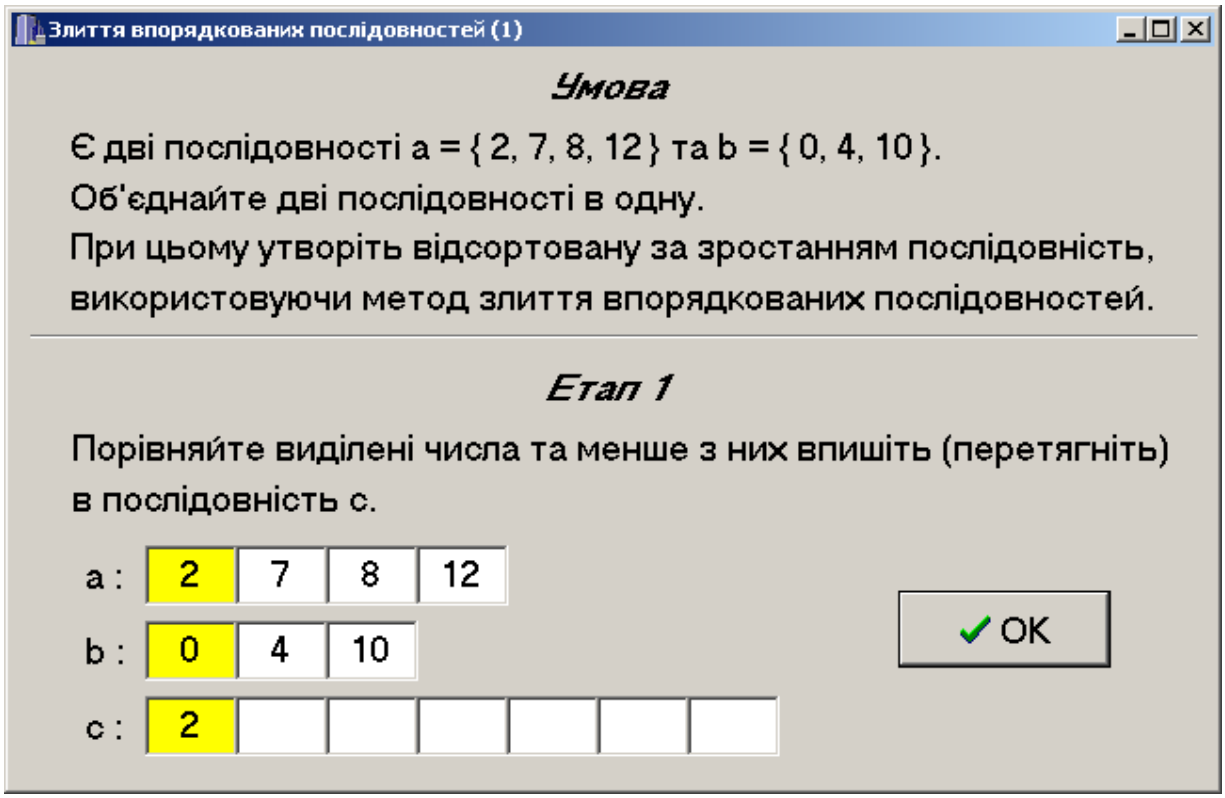

Рисунок 4.4 – Етап 1 з помилковою відповіддю

Користувач може ввести число, а може перетягнути інформацію з однієї із виділених клітинок.

У випадку помилки (рис. 4.4) з'являється пояснення, в чому полягає помилка (рис. 4.5). Після чого користувач повинен виправити помилку.

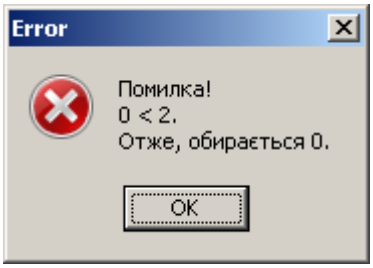

Рисунок 4.5 – Пояснення помилки

У випадку вірної відповіді (рис. 4.6) з'являється підтверджуюче повідомлення (рис. 4.7). Далі поточний елемент в тій послідовності, з якої був обраний елемент, змінюється на наступний. Про це видається відповідне повідомлення (рис. 4.8).

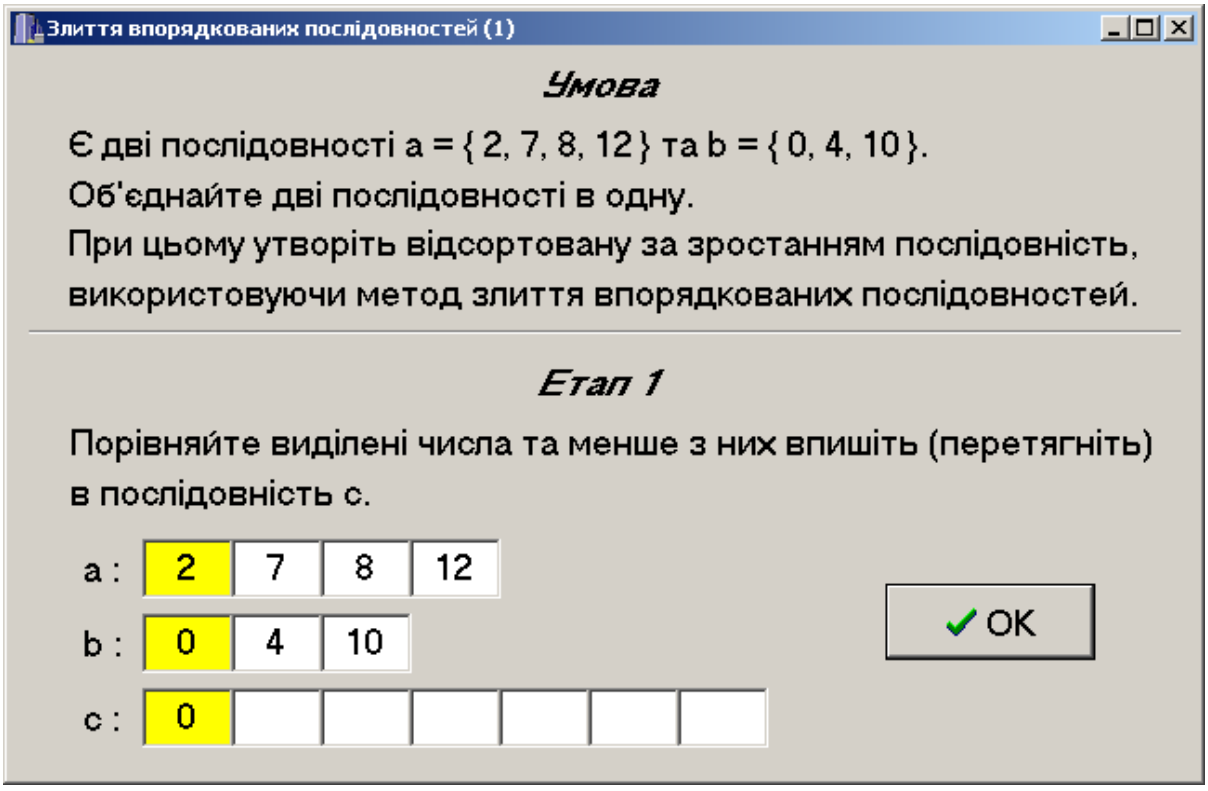

Рисунок 4.6 – Етап 1 з вірною відповіддю

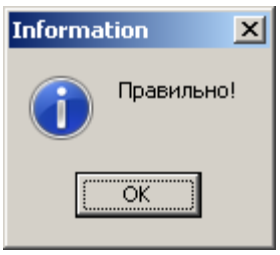

Рисунок 4.7 – Підтвердження вірності

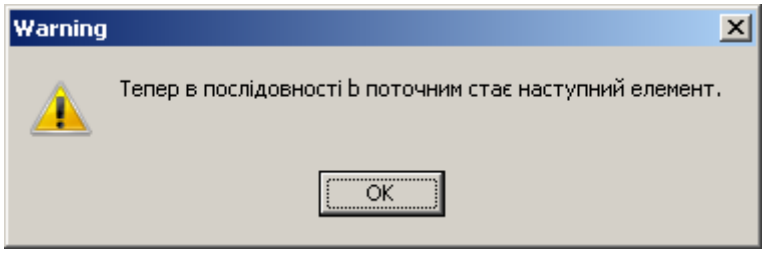

Рисунок 4.8 – Повідомлення про зміну поточного елемента

Наступні кроки програми працюють аналогічно (рис. 4.9-4.27).

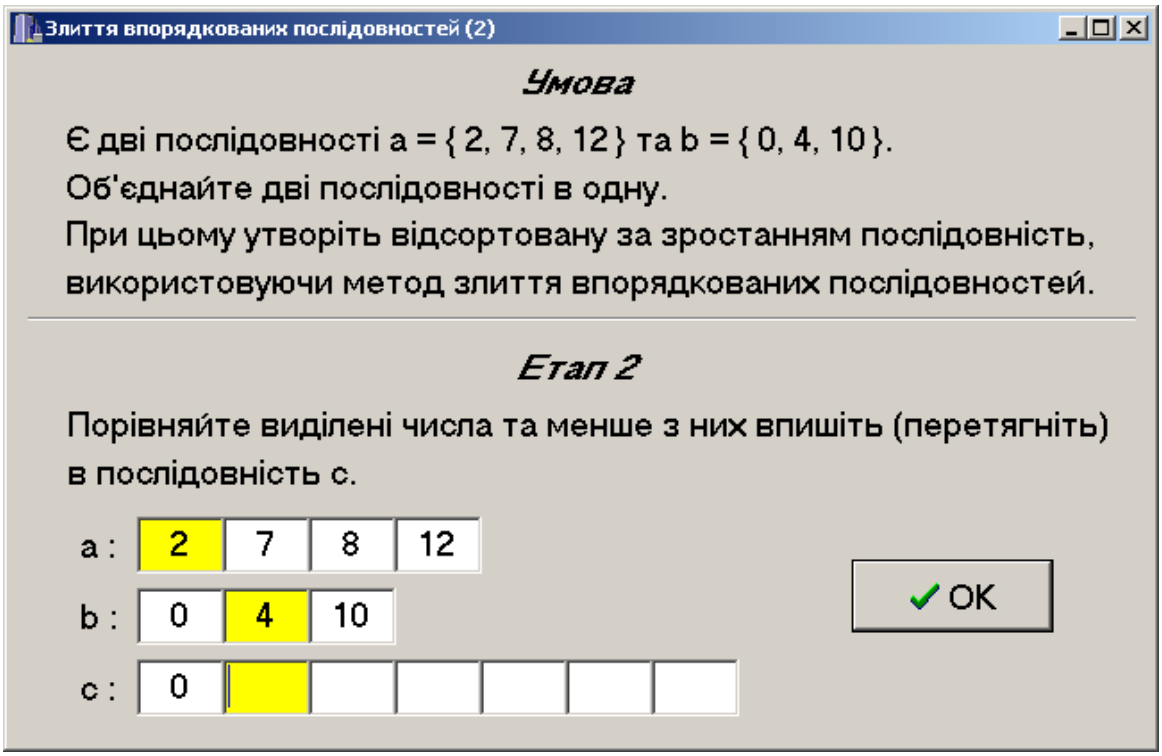

Рисунок 4.9 – Етап 2

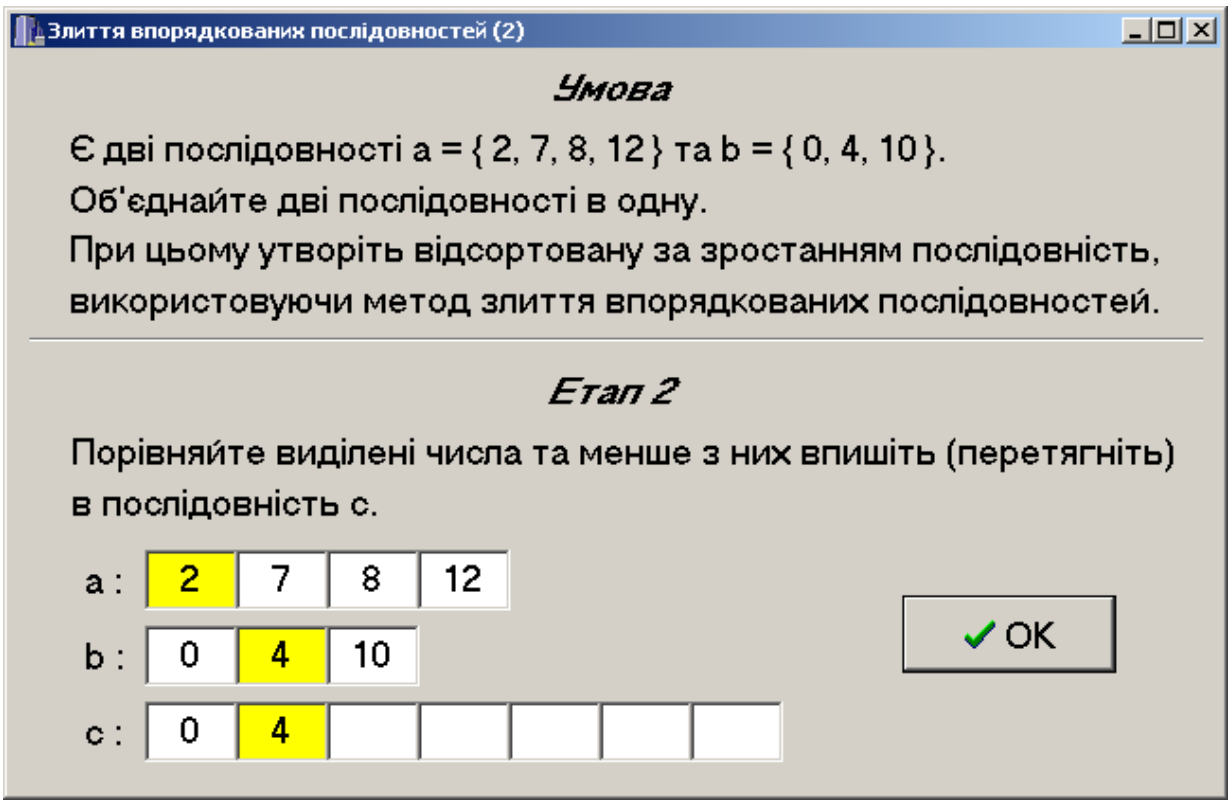

Рисунок 4.10 – Етап 2 з помилковою відповіддю

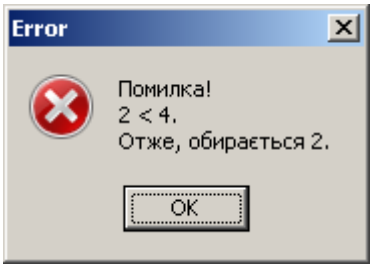

Рисунок 4.11 – Пояснення помилки

Після виконання шостого кроку програми в послідовності *b* усі елементи виявляються перебраними. Це означає, що у послідовність *c* вписується елементи, що залишились в послідовності *a* (рис. 4.23-4.25).

Після проходження тренінгу з’являються відповідні повідомлення (рис. 4.26-4.27).

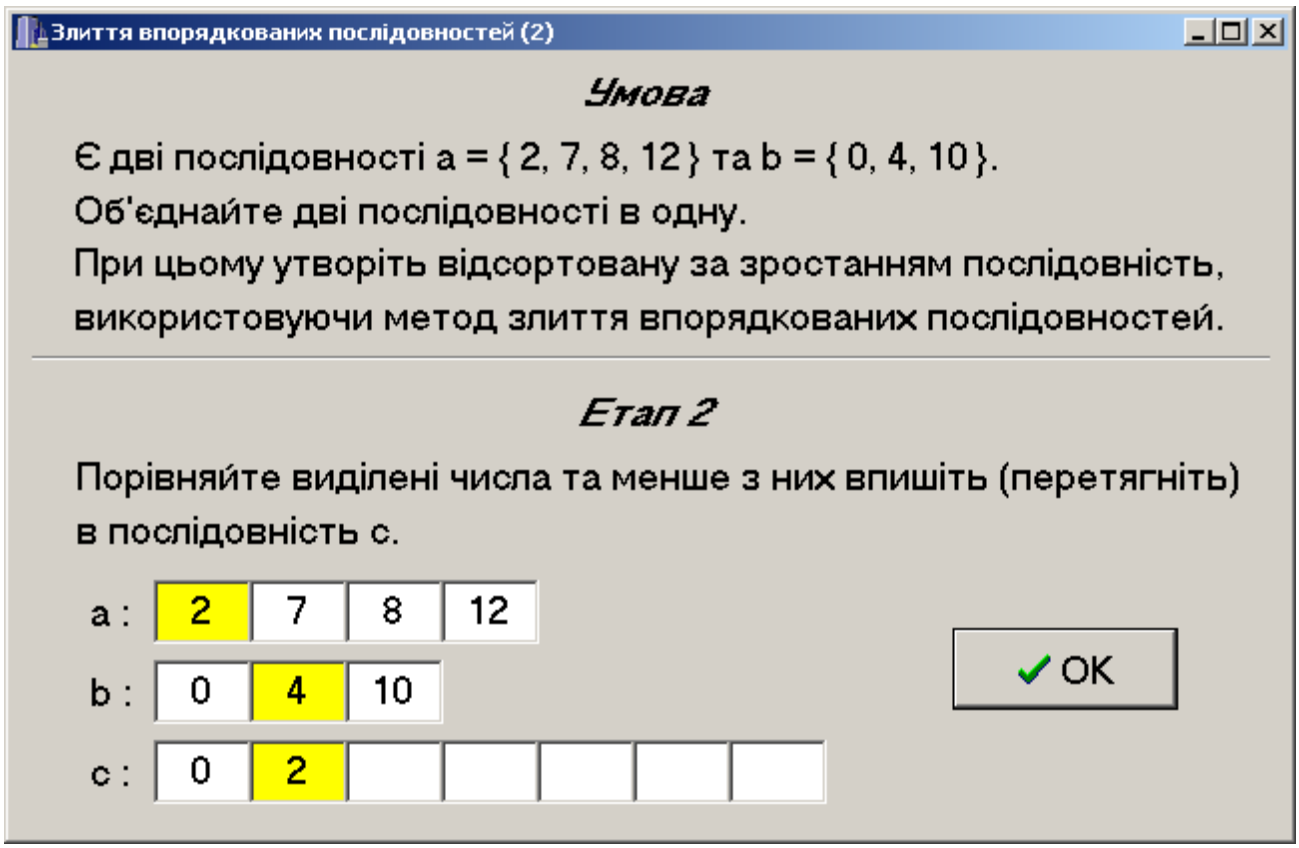

Рисунок 4.12 – Етап 2 з вірною відповіддю

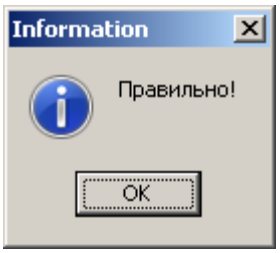

Рисунок 4.13 – Підтвердження вірності

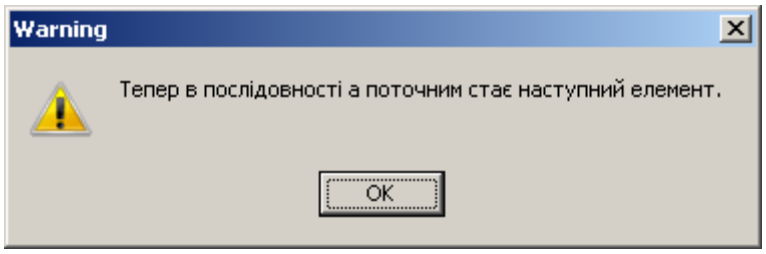

Рисунок 4.14 – Повідомлення про зміну поточного елемента

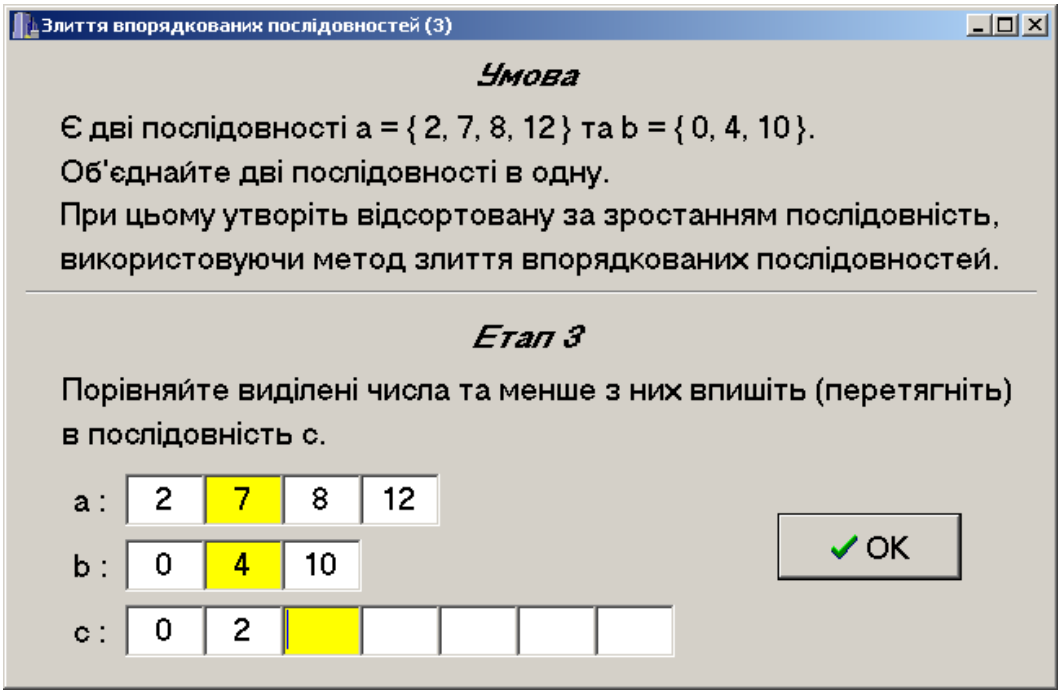

Рисунок 4.15 – Етап 3

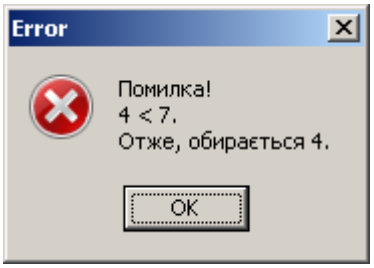

Рисунок 4.16 – Пояснення помилки

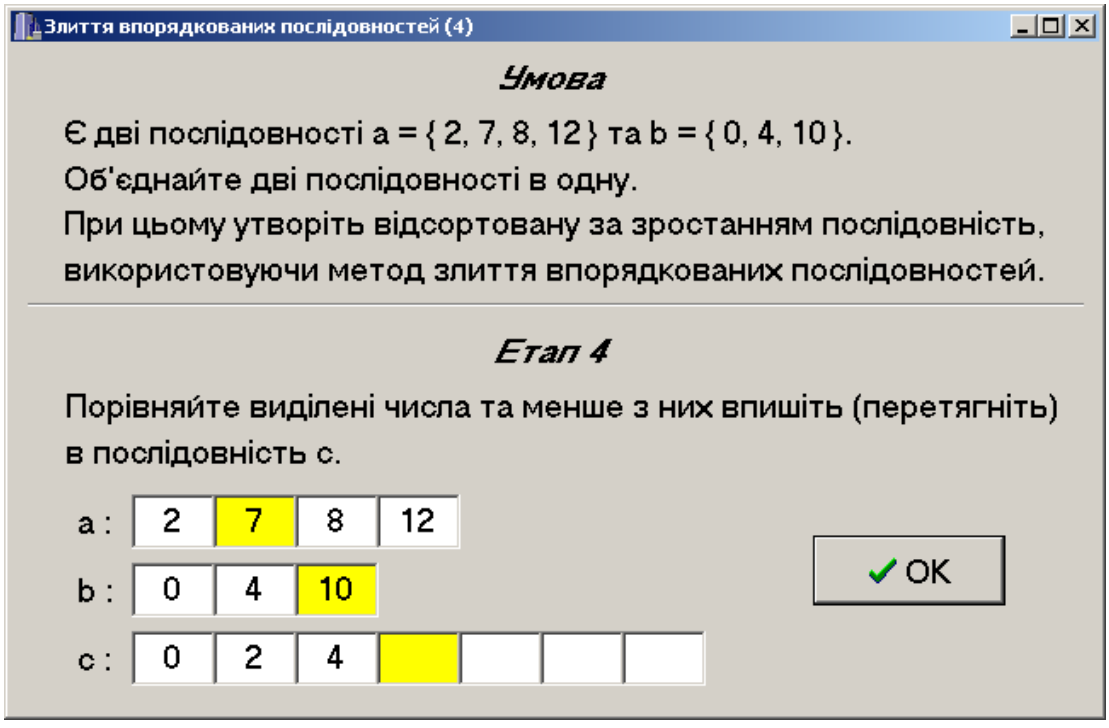

Рисунок 4.17 – Етап 4

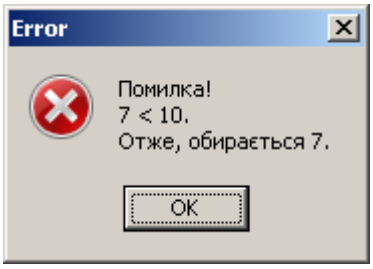

Рисунок 4.18 – Пояснення помилки

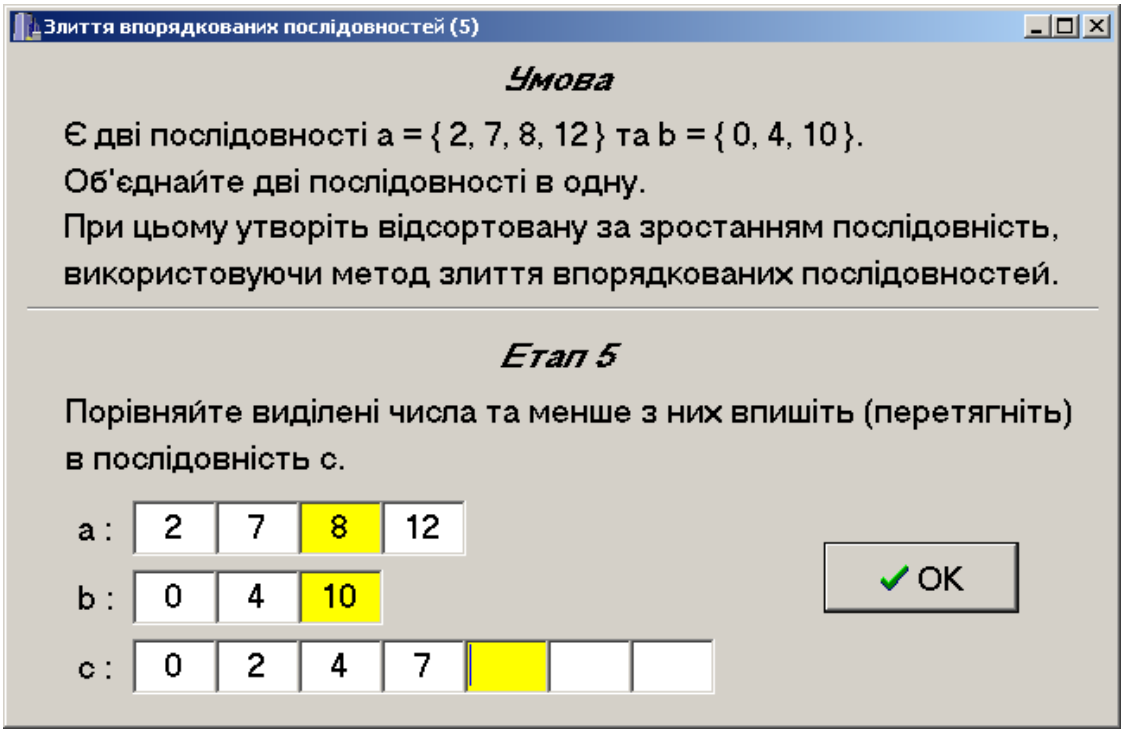

Рисунок 4.19 – Етап 5

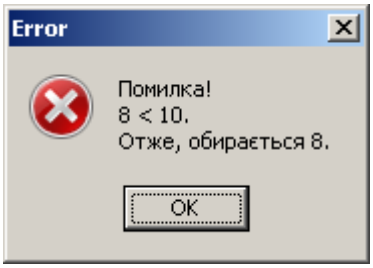

Рисунок 4.20 – Пояснення помилки

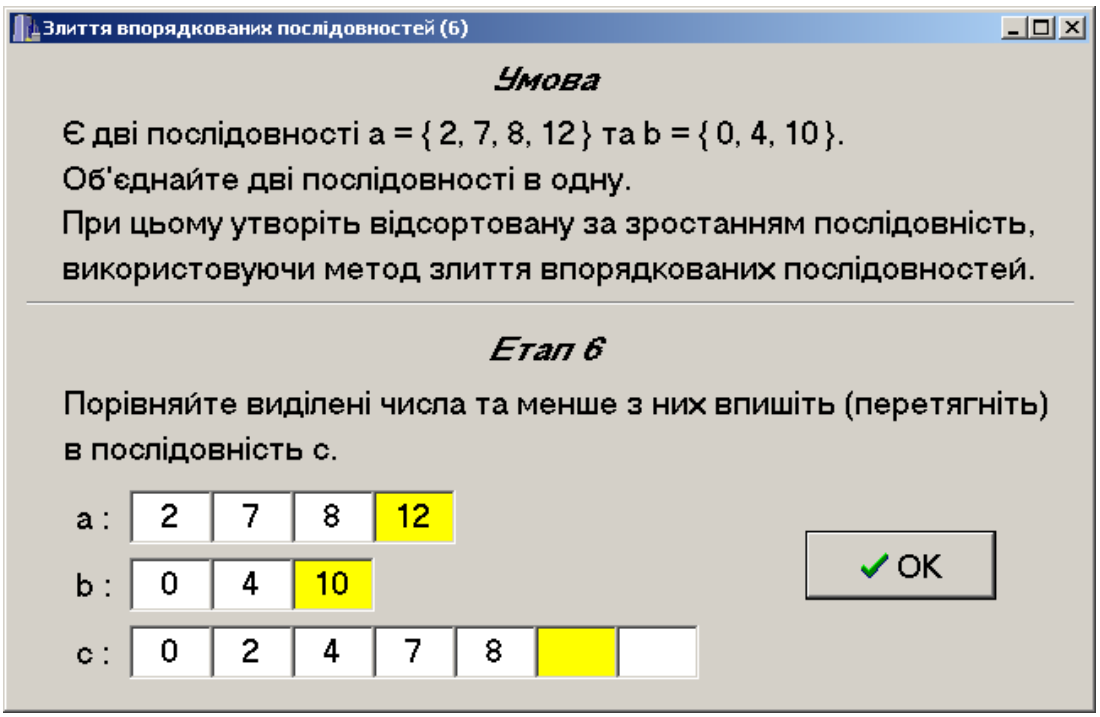

Рисунок 4.21 – Етап 6

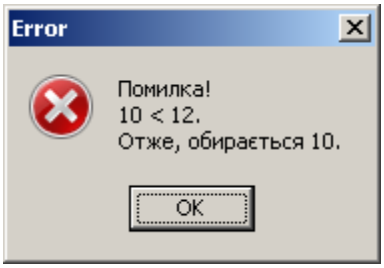

Рисунок 4.22 – Пояснення помилки

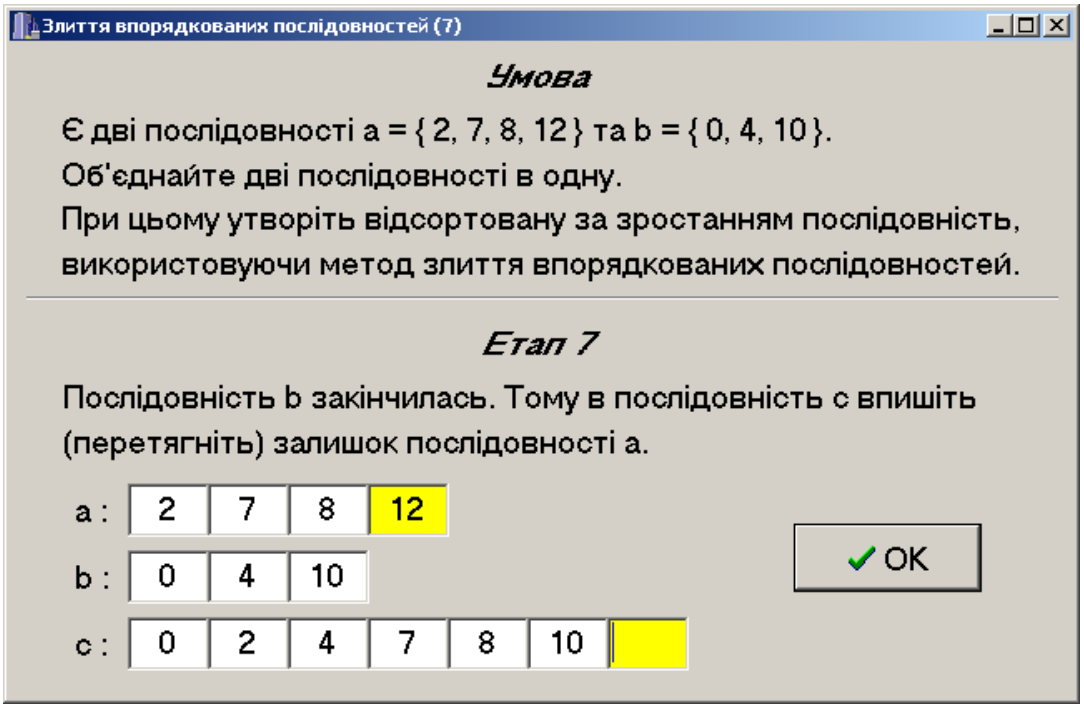

Рисунок 4.23 – Етап 7

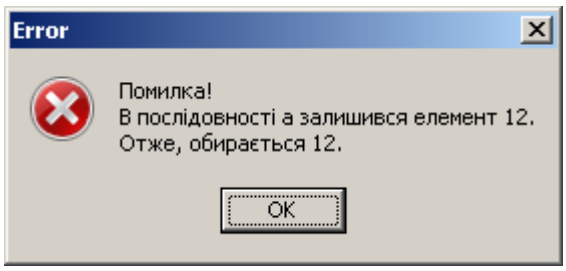

Рисунок 4.24 – Пояснення помилки

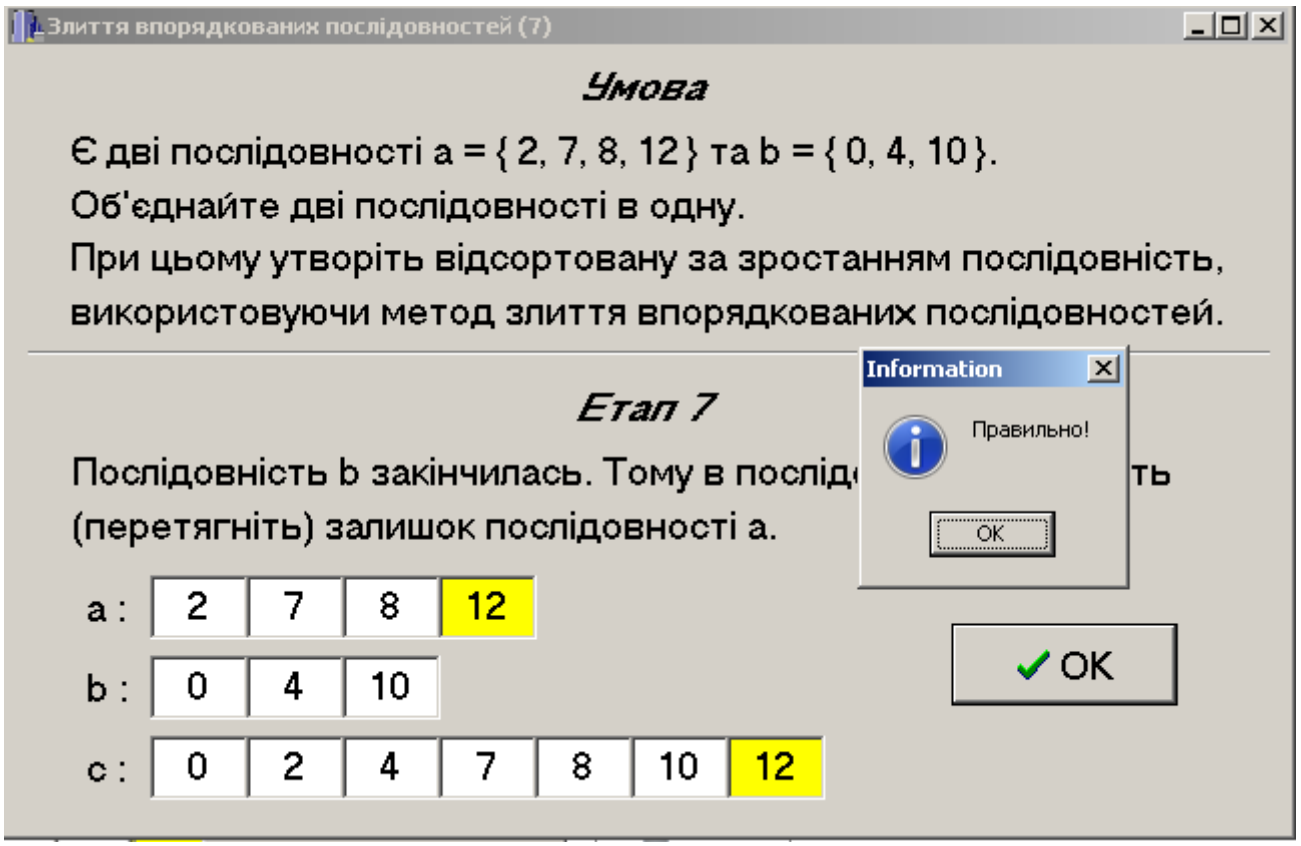

Рисунок 4.25 – Етап 7 з вірною відповіддю

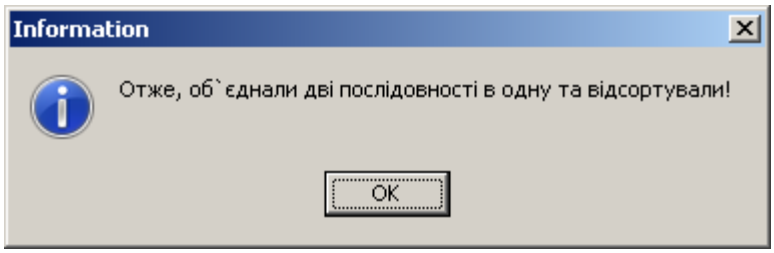

Рисунок 4.26 – Висновок

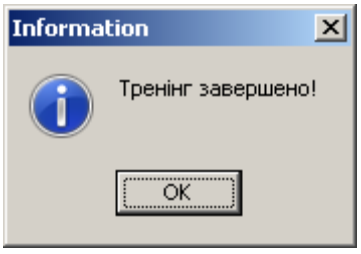

Рисунок 4.27 – Повідомлення про кінець

ВИСНОВКИ

В бакалаврський роботі досліджено тему «Злиття впорядкованих послідовностей». Основним теоретичним матеріалом була інформація з дистанційного курсу Полтавського університету економіки та торгівлі «Алгоритми та структури даних».

Було здійснено огляд візуалізацій сортування злиттям, оскільки метод «злиття впорядкованих послідовностей» є головною ланкою при сортуванні злиттям.

На базі прикладу з дистанційного курсу створено перший приклад тренажеру: був розроблений алгоритм, блок-схема алгоритму та створена програмна реалізація.

Другий приклад для тренажеру було прораховано самостійно. За другого прикладу також був розроблений алгоритм та створена програмна реалізація.

Програма є працюючою та перевіреною на відсутність помилок.

В пояснювальній записці описано, як створювалась програма, та як програма працює.

Тренажер буде передано у сектор розробки електронних ресурсів Полтавського університету економіки та торгівлі для впровадження у дистанційний курс кафедри математичного моделювання та соціальної інформатики «Алгоритми та структури даних».

Результати бакалаврської роботи пройшли обговорення на науково-практичному семінарі «Комп’ютерні науки та прикладна математика»-2020, що проходив на базі кафедри математичного моделювання та соціальної інформатики Полтавського університету економіки та торгівлі.

Були опубліковані тези у збірнику матеріалів цього семінару [13].

ЛІТЕРАТУРА

1. Ємець Ол-ра О. Дистанційний курс Полтавського університету економіки та торгівлі «Алгоритми та структури даних» для студентів спеціальностей «Комп'ютерні науки та інформаційні технології», «Комп'ютерні науки» / Ол-ра О. Ємець. – [Електронний ресурс].

2. Соколов О. Ю. Інформатика для інженерів / О. Ю. Соколов, І. Т. Зарецька, Г. М. Жолткевич, О. В. Ярова. – Харків: Факт, 2006. – 424 с.

3. Ахо А. Структуры данных и алгоритмы / А. Ахо, Дж. Хопкрофт, Дж. Ульман. – М.: Вильямс, 2003. – 384 с.

4. Бакнелл Джулиан М. Фундаментальные алгоритмы и струкутры даннях в Delphi / Джулиан М. Бакнелл. – СПб.: ООО «ДиаСофтЮП», 2003. – 560 с.

5. Вирт Н. Алгоритмы и структуры данных / Н. Вирт. – СПб: Невский диалект, 2001. – 352 с.

6. Кнут Д. Искусство программирования. Т. 3. Сортировка и поиск / Д. Кнут. – М.: Изд. дом «Вильямс», 2000. – 824 с.

7. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Риверст. – М. МНЦО, 2000. – 960 с.

8. Красиков И. В. Алгоритмы. Просто как дважды два / И. В. Красиков, И. Е. Красикова. – М.: Эксмо, 2006. – 256 с.

9. Седжвик Р. Фундаментальные алгоритмы на С++: Части 1-4: Анализ. Структуры данных. Сортировка. Поиск / Р. Седжвик. – К.: Из-во «Диасофт», 2001. – 688 с.

10. Data Structure Visualizations (Візуалізації структур даних університету Сан-Франциско). – [Електронний ресурс]. – Режим доступу: http://www.cs.usfca.edu/~galles/visualization/Algorithms.html.

11. Сортування злиттям (візуалізація за допомогою німецького національного танця). – [Електронний ресурс]. – Режим доступу: http://www.youtube.com/watch?v=XaqR3G_NVoo.

12. Сортування. Візуалізація точками. – [Електронний ресурс]. – Режим доступу: https://youtu.be/zcO8uxg_Spw.

13. Кулинич М. К. Тренажер «Злиття впорядкованих послідовностей» / К. М. Кулинич, Є. М. Ємець, О. О. Ємець // Комп'ютерні науки і прикладна математика (КНіПМ-2020): матеріали наук.-практ. семінару. Випуск 5. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2020. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/8268.

14. Ємець О. О. Про розробку тренажерів для дистанційних курсів кафедрою ММСІ ПУЕТ / О. О. Ємець // Інформатика та системні науки (ІСН-2015): матеріали VI Всеукр. наук.-практ. конф. за міжн. участю (м. Полтава, 19-21 березня 2015 р.) / за ред. Ємця О. О. – Полтава: ПУЕТ, 2015. – С. 152-161. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/2488.

15. Хрупа О. І. Розробка програмного забезпечення з теми «Турнірне сортування» дистанційного навчального курсу «Алгоритми та структури даних» / О. І. Хрупа, Ол-ра О. Ємець // Комп'ютерні науки і прикладна математика (КНіПМ-2019): матеріали наук.-практ. семінару. Випуск 3. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2019. – С. 43-45. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/7039.

16. Чуб О. І. Тренажер «Рекурсивні алгоритми» / О. І. Чуб, О.О. Ємець // Комп'ютерні науки і прикладна математика (КНіПМ-2019): матеріали наук.-практ. семінару. Випуск 4. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2019. – С. 16-19. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/7456.

17. Ємець О. О. Методичні рекомендації до виконання бакалаврської роботи для студентів спеціальності 122 «Комп`ютерні науки та інформаційні технології» освітня програма «Комп`ютерні науки» галузь знань – 12 «Інформаційні технології» / О. О. Ємець. – Полтава: ПУЕТ, 2017. – 71 с